MULTIMEDIA UNIVERSITY

STUDENT ID NO

MULTIMEDIA UNIVERSITY

FINAL EXAMINATION

TRIMESTER 1, 2018/2019

ECE4246 – ADVANCED OBJECT-ORIENTED DESIGN WITH JAVA

(All Sections / Groups)

20 OCTOBER 2018
2:30 P.M – 4:30 P.M.
(2 Hours)

INSTRUCTIONS TO STUDENT

1. This examination paper consists of FIVE (5) pages, including the cover page, with THREE (3) questions only.
2. Answer ALL questions. All questions carry 25 marks and the distribution of the marks for each question is given.
3. This examination is OPEN-BOOK. Allowed material: Textbook and/or bound notes only.
4. Please print all your answers in the Answer Booklet provided.

Question 1

Create a Java application that can convert weight in the units of stones and pounds to kilograms (kg). Given:

1 stone = 6.35029kg
1 pound = 0.45359kg

(a) Write a class, `ConversionPanel`, which is a `JPanel` class, and contains a text field for input, a combo box for selection of input unit (stones or pounds), two buttons ("Convert" and "Reset"), and a label for displaying the output value (in kg). The output label displays an empty string at the start of execution. See Figure Q1a and Figure Q1b.

[10 marks]

When the button "Convert" is clicked, obtain the input entered in the text field and perform the right conversion based on the currently-selected input unit (stones or pounds). Display the resulting value in kg. See Figure Q1c. When the "Reset" button is clicked, restore all GUI elements to the starting state shown in Figure Q1a.

[8 marks]

Handle all exceptions that may occur.

[3 marks]

(b) Write an application class, `ConversionTool`, to display an instance of `ConversionPanel` in a frame.

[4 marks]

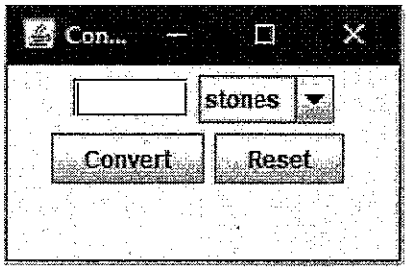

Figure Q1a: GUI elements

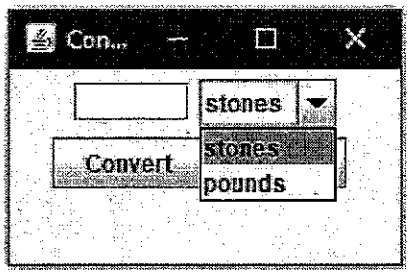

Figure Q1b: Combo box items

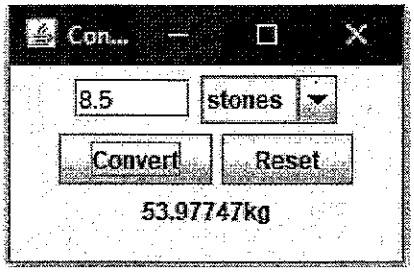

Figure Q1c: Conversion result

Continued...

Question 2

(a) Write an interface, `Rateable`, which has two methods: `setRating(double)`, which does not return any value, and `getRating()`, which returns a `double` value

[2 marks]

(b) Write an abstract class, `UniCitizen`, which implements the interface `Rateable`. This class has an instance variable, `name` (type `String`).

- Provide a constructor, mutator and accessor methods
- Provide a `toString` method. This method returns the value of its instance variable.
- Override the `equals` method of `Object` class to determine the equality between two `UniCitizen` objects.

[7 marks]

(c) Write a class, `Educator`, which inherits from `UniCitizen`. Declare an instance variable, `evaluation` (type `double`), which represents the educator's teaching evaluation and ranges from 0 to 5.0, inclusive.

- Provide a constructor.
- Implement superclass abstract methods `setRating` and `getRating`, with "rating" being the value of `evaluation`.
- Override the superclass method `equals`.

[7 marks]

(d) Write a class, `Learner`, which inherits from `UniCitizen`. This class has two instance variables: `educator` (type `Educator`) and `cpga` (type `double`). The value of `cgpa` ranges from 0 to 4.0, inclusive.

- Provide a constructor. Implement superclass abstract methods `setRating` and `getRating`, with "rating" being the value of `cpga`.
- Override the superclass methods `equals`.

[7 marks]

(e) Draw a Unified Modeling Language (UML) class diagram, displaying only the class names (**DO NOT** list any of the members of the classes) to show the relationships among `Rateable`, `UniCitizen`, `Educator`, and `Learner`.

[2 marks]

Continued...

Question 3

Figure Q3a shows the code for the interface **ScoredChallenge**.

```
public interface ScoredChallenge
{    public void runChallenge ();
     public int getScore ();
}
```

Figure Q3a: Interface **ScoredChallenge**

(a) Write a class, **TypingSpeed**, which implements **ScoredChallenge**. This class times a user's typing speed and awards a score based on it. It has instance variables **phrase** (type **String**) and **duration** (type **long**). Provide a constructor which accepts a **String** argument and initializes phrase using the argument. Variable **duration** is initialized to -1. Implement all methods from the interface **ScoredChallenge**:

- **runChallenge**: Display **phrase** and instruct the user to type it. Time the user by counting the seconds between the prompt being displayed and the phrase being input by the user. (*Hint*: The method **System.currentTimeMillis()** returns the current time in milliseconds as a **long** value). Store the user's typing duration in the instance variable **duration**.

- **getScore**: Determine the score based on the user's typing speed and return it. If the user typed and input the phrase in 5 seconds or less, he / she is considered fast and the score would be 3. Otherwise, the score is 1.

[10 marks]

(b) Write a class, **CountingSkill**, which implements **ScoredChallenge**. This class tests a user's counting accuracy and awards a score if the count is correct. It has instance variables **count** (type **int**) and **isCorrect** (type **boolean**). Provide a constructor which initializes **count** to 0 and **isCorrect** to **false**. Implement all methods from the interface **ScoredChallenge**:

- **runChallenge**: Generate a random value ranging from 1 to 10, inclusive, and assign it to **count**. Display a number of 1's equal to **count** value, and a "=" at the end of the expression. For example, if **count** value is 4, display:

```
1+1+1+1=
```

Continued...

Allow the user to count the number of 1's displayed and input the answer. If the user's answer equals `count` (number of 1's displayed), set `isCorrect` to `true`. Otherwise, set `isCorrect` to `false`.

- `getScore`: Provide the user's score based on whether he/she had counted the number of 1's displayed correctly. Return a score value of 2 for `isCorrect` being `true`, and 0 otherwise.

[10 marks]

(c) Identify the design pattern implemented in the design of `ScoredChallenge`, `TypingSpeed`, and `CountingSkill`. Briefly describe the context, in terms of the said interface / classes, and its corresponding solution.

[5 marks]

***Note*: <u>DO NOT</u> write the driver class.**

End of Paper